Lecture 2:

- Linear Regression
- Gradient Descent
- Non-linear basis functions
LINEAR REGRESSION

MOTIVATION
Why Linear Regression?

- Regression = Prediction of real-valued outputs

- Simplest regression algorithm
  - Easy, and fast
  - Benchmark algorithm

- Mathematical Concepts introduced
  - Data format and Matrix notation
  - Minimizing a cost function: gradient descent
  - Non-linear features and basis functions
Examples: (linear) regression application

- Social science: relationship between data
- Brain computer interfaces
- Neuroprosthetic control
Examples: (linear) regression application

• Social science: relationship between data
• Brain computer interfaces
• Neuroprosthetic control
LINEAR REGRESSION WITH ONE INPUT
Linear regression with one input

\[ \langle x^{(1)}, y^{(1)} \rangle \ldots \langle x^{(m)}, y^{(m)} \rangle \]

Training set

Learning algorithm

\[ h_{\theta}(x) = \theta_0 + \theta_1 \cdot x \]

Hypothesis

Parameters \( \theta = (\theta_0, \theta_1) \)

Test input

\( x \)

"Hypothesis“ \( h \)

Prediction

\[ 45 \quad 50 \quad 55 \quad 60 \]

\[ 170 \quad 175 \quad 180 \quad 185 \]

knee height

body height
A regression problem

• We want to learn to predict a **person’s height** based on his/her **knee height** and/or **arm span**

• This is useful for patients who are **bed bound** or in a wheelchair and cannot stand to take an accurate measurement of their height

<table>
<thead>
<tr>
<th>Knee Height [cm]</th>
<th>Arm span [cm]</th>
<th>Height [cm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>166</td>
<td>171</td>
</tr>
<tr>
<td>56</td>
<td>172</td>
<td>175</td>
</tr>
<tr>
<td>52</td>
<td>174</td>
<td>168</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Example Data

<table>
<thead>
<tr>
<th>Knee height [cm]</th>
<th>Arm span [cm]</th>
<th>Height [cm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>166</td>
<td>171</td>
</tr>
<tr>
<td>56</td>
<td>172</td>
<td>175</td>
</tr>
<tr>
<td>52</td>
<td>174</td>
<td>168</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

$m=30$ data points
# Example Data

<table>
<thead>
<tr>
<th>Knee Height [cm]</th>
<th>Arm span [cm]</th>
<th>Height [cm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>166</td>
<td>171</td>
</tr>
<tr>
<td>56</td>
<td>172</td>
<td>175</td>
</tr>
<tr>
<td>52</td>
<td>174</td>
<td>168</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Linear regression with one input

Which hypothesis is better?

In what sense is it better?

Hypothesis

\[ h_\theta(x) = \theta_0 + \theta_1 \cdot x \]

Parameters

\[ \theta = (\theta_0, \theta_1) \]
Formalization of problem

- Given $m$ training examples
  \[ \langle x^{(1)}, y^{(1)} \rangle \ldots \langle x^{(m)}, y^{(m)} \rangle \]

- Goal: learn parameters
  \[ \theta = (\theta_0, \theta_1) \]
  such that
  \[ h_\theta(x^{(i)}) = \theta_0 + \theta_1 \cdot x^{(i)} \approx y^{(i)} \]
  for all training examples $i=1\ldots30$. 

<table>
<thead>
<tr>
<th>Knee Height [cm]</th>
<th>Height [cm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>171</td>
</tr>
<tr>
<td>56</td>
<td>175</td>
</tr>
<tr>
<td>52</td>
<td>168</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

$m=30$ data points
Least Squares Objective

- Minimize Error

$$J(\theta_0, \theta_1) = \left( h_\theta \left(x^{(i)}\right) - y^{(i)} \right)^2$$

$$\theta_0 = 150$$
$$\theta_1 = 0.6$$
Least Squares Objective

- Minimize Error

\[ J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta(x^{(i)}) - y^{(i)} \right)^2 \]

\[ \theta_0 = 150 \]
\[ \theta_1 = 0.6 \]
\[ J(\theta_0, \theta_1) = 10.77 \]
Least Squares Objective

- Minimize Error

\[ J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left( h_{\theta}(x^{(i)}) - y^{(i)} \right)^2 \]

\[ \theta_0 = 140 \]
\[ \theta_1 = 0.75 \]
\[ J(\theta_0, \theta_1) = 5.94 \]
Cost function illustrated

\[ J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta(x^{(i)}) - y^{(i)} \right)^2 \]

Properties of cost function:

- **Quadratic** function
- **Convex** "Bowl"-shaped
- Unique local and global minimum (under "regular" conditions)
Minimizing the cost

• Two ways to find the parameters $\theta = (\theta_0, \theta_1)$ minimizing

$$J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta \left( x^{(i)} \right) - y^{(i)} \right)^2$$

• Gradient descent
• Direct analytical solution
  (setting derivatives = 0)
EXCURSUS: GRADIENT DESCENT
Descending in the steepest direction

Gradient descent on some arbitrary cost function $J(\theta_0, \theta_1)$ ...
Gradient descent algorithm

- Repeat until convergence

\[ \theta_j := \theta_j - \eta \cdot \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \]

(simultaneously updating \( \theta_0 \) and \( \theta_1 \))

- negative gradient = descent
- learning rate ("eta")
- partial derivative of \( J(\theta_0, \theta_1) \) with respect to \( \theta_j \)
Gradient is orthogonal to contour lines

\[ J(\theta_0, \theta_1) \]

A contour line is a line along which
\[ J(\theta_0, \theta_1) = \text{const} \]
Potential issues with gradient descent

- May get stuck in local minima
- Learning rate too small: slow convergence
- Learning rate too large: oscillations, divergence

\[\eta\] too small

\[\eta\] too large
LINEAR REGRESSION WITH GRADIENT DESCENT

(ONE INPUT)
Application of gradient descent

- **Linear regression cost**

\[ J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta \left( x^{(i)} \right) - y^{(i)} \right)^2 \]

\[ h_\theta(x) = \theta_0 + \theta_1 \cdot x \]

- **Gradient descent**

\[ \theta_j := \theta_j - \eta \cdot \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \]

(simultaneous update)

\[ \theta_0 := \theta_0 - 2\eta \cdot \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta \left( x^{(i)} \right) - y^{(i)} \right) \]

learning rate

\[ \theta_1 := \theta_1 - 2\eta \cdot \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta \left( x^{(i)} \right) - y^{(i)} \right) \cdot x^{(i)} \]

(simultaneous update)

"error"

"input"
Predicting height from knee height

- Optimal fit to training data

\[
\theta_0 = 137.4 \\
\theta_1 = 0.8
\]
LINEAR REGRESSION
MORE GENERAL FORMULATION: MULTIPLE FEATURES
Multiple inputs (features)

\[ \begin{array}{cccc}
\text{Knee Height} & \text{Arm span} & \text{Age} & \text{Height} \\
{x_1} & {x_2} & {x_3} & y \\
50 & 166 & 32 & 171 \\
56 & 172 & 17 & 175 \\
52 & 174 & 62 & 168 \\
\ldots & \ldots & \ldots & \ldots \\
\end{array} \]

\[ m \]

\[ n = 3 \]

- Notation:
  - \( m \) ... number of training examples
  - \( n \) ... number of features

\[ \mathbf{x}^{(i)} \] ... input features of \( i \)th training example (vector-valued)

\[ x_j^{(i)} \] ... value of feature \( j \) in \( i \)th training example

\[ \mathbf{x}^{(2)} = \begin{pmatrix} 56 \\ 172 \\ 17 \end{pmatrix} \]

\[ x_3^{(2)} = 17 \]
Linear hypothesis

- Hypothesis (one input):
  \[ h_\theta(x) = \theta_0 + \theta_1 \cdot x \]

- Hypothesis (multiple input features):
  \[ h_\theta(x) = \theta_0 + \theta_1 \cdot x_1 + \cdots + \theta_n \cdot x_n \]

Example: \( h(x) = 50 + 0.5 \cdot \text{kneeheight} + 0.3 \cdot \text{armspan} + 0.1 \cdot \text{age} \)

- More compact notation:
  \[ h_\theta(x) = x^T \theta \]

Introduce \( x_0 = 1 \)

Why? Notation convenience!
Multiple inputs (features) revisited

- Notation:
  - $m$ ... number of training examples
  - $n$ ... number of features

- $\mathbf{x}^{(i)}$ ... input features of $i$'th training example (vector-valued)
- $x_{j}^{(i)}$ ... value of feature $j$ in $i$'th training example

<table>
<thead>
<tr>
<th>$\mathbf{x}_0$</th>
<th>Knee Height</th>
<th>Arm span</th>
<th>Age</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>166</td>
<td>32</td>
<td>171</td>
</tr>
<tr>
<td><strong>1</strong></td>
<td><strong>56</strong></td>
<td><strong>172</strong></td>
<td><strong>17</strong></td>
<td>175</td>
</tr>
<tr>
<td>1</td>
<td>52</td>
<td>174</td>
<td>62</td>
<td>168</td>
</tr>
<tr>
<td>1</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- $n = 3$

$$\mathbf{x}^{(2)} = \begin{pmatrix}
1 \\
56 \\
172 \\
17
\end{pmatrix}$$

$$x_0^{(2)} = 1$$

$$x_3^{(2)} = 17$$
Matrix and vector notation

<table>
<thead>
<tr>
<th>$x_0$</th>
<th>Knee Height</th>
<th>Arm span</th>
<th>Age</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>166</td>
<td>32</td>
<td>171</td>
</tr>
<tr>
<td>1</td>
<td>56</td>
<td>172</td>
<td>17</td>
<td>175</td>
</tr>
<tr>
<td>1</td>
<td>52</td>
<td>174</td>
<td>62</td>
<td>168</td>
</tr>
</tbody>
</table>

$$X = \begin{pmatrix}
1 & 50 & 166 & 32 \\
1 & 56 & 172 & 17 \\
1 & 52 & 174 & 62
\end{pmatrix}$$

$$y = \begin{pmatrix}
171 \\
175 \\
168
\end{pmatrix}$$

$$\mathbf{x}^{(i)} = \begin{pmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{pmatrix}$$

$$\mathbf{X} = \begin{pmatrix}
(\mathbf{x}^{(1)})^T \\
(\mathbf{x}^{(2)})^T \\
\vdots \\
(\mathbf{x}^{(m)})^T
\end{pmatrix}$$

$$\mathbf{y} = \begin{pmatrix}
y^{(1)} \\
y^{(2)} \\
\vdots \\
y^{(m)}
\end{pmatrix}$$

features of $i$'th training example $(n+1) \times 1$

design matrix $m \times (n+1)$

output/target vector $m \times 1$
LINEAR REGRESSION WITH GRADIENT DESCENT
(GENERAL FORMULATION)
Linear regression problem statement

• Hypothesis: \( h_\theta (x) = x^T \theta \)

• Cost function: \( J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta (x^{(i)}) - y^{(i)} \right)^2 \)

Goal is to find parameters which minimize the cost
Gradient descent (multiple features)

with one input feature:

\[
\theta_0 := \theta_0 - 2\eta \cdot \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta \left( x^{(i)} \right) - y^{(i)} \right)
\]

\[
\begin{aligned}
\theta_1 := \theta_1 - 2\eta \cdot \frac{1}{m} & \sum_{i=1}^{m} \left( h_\theta \left( x^{(i)} \right) - y^{(i)} \right) \cdot x^{(i)} \\
\end{aligned}
\]

(with simultaneous update)

with \( n \) input features:

\[
\begin{aligned}
\theta_j := \theta_j - 2\eta \cdot \frac{1}{m} & \sum_{i=1}^{m} \left( h_\theta \left( x^{(i)} \right) - y^{(i)} \right) \cdot x_j^{(i)} \\
\end{aligned}
\]

(with simultaneous update for \( j=0\ldots n \))

For \( j=0 \): define for convenience \( x_0^{(i)} = 1 \)
LINEAR REGRESSION
ANALYTICAL SOLUTION
Analytical solution

• Set all partial derivatives of cost function $J(\theta) = 0$

• Solving system of linear equations yields:

$$
\theta^* = \left( X^T X \right)^{-1} X^T y
$$

Moore-Penrose Pseudoinverse of $X$

$x$ ... design matrix

$y$ ... output/target vector

• Note: This analytical solution requires that columns of $X$ are linearly independent ("regular" conditions)
Example: analytical solution applied to problem with one input

<table>
<thead>
<tr>
<th>Knee Height [cm]</th>
<th>Height [cm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>171</td>
</tr>
<tr>
<td>56</td>
<td>175</td>
</tr>
<tr>
<td>52</td>
<td>168</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
Example: analytical solution applied to problem with one input

<table>
<thead>
<tr>
<th>Knee Height [cm]</th>
<th>Height [cm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>171</td>
</tr>
<tr>
<td>56</td>
<td>175</td>
</tr>
<tr>
<td>52</td>
<td>168</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\[ X = \begin{pmatrix} 1 & 50 \\ 1 & 56 \\ 1 & 52 \\ \vdots \end{pmatrix} \quad y = \begin{pmatrix} 171 \\ 175 \\ 168 \\ \vdots \end{pmatrix} \]

\[ X^T X = \begin{pmatrix} 30 & 1577 \\ 1577 & 83222 \end{pmatrix} \quad 2 \times 2 \]

\[ (X^T X)^{-1} = \begin{pmatrix} 7.994 & -0.152 \\ -0.152 & 0.003 \end{pmatrix} \quad 2 \times 2 \]

\[ X^T y = \begin{pmatrix} 5383 \\ 283210 \end{pmatrix} \quad 2 \times 1 \]

\[ \theta^* = (X^T X)^{-1} X^T y \]

\[ = \begin{pmatrix} 137.4 \\ 0.8 \end{pmatrix} \]
Predicting height from knee height

\[ \theta_0 = 137.4 \]
\[ \theta_1 = 0.8 \]

\[
\theta^* = \left( X^T X \right)^{-1} X^T y
\]
\[
= \begin{pmatrix} 137.4 \\ 0.8 \end{pmatrix}
\]
<table>
<thead>
<tr>
<th>Gradient descent</th>
<th>Analytical solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Need to choose learning rate $\eta$</td>
<td>• No need to choose $\eta$</td>
</tr>
<tr>
<td>• Iterative algorithm (needs many iterations to converge)</td>
<td>• Direct solution (no iteration)</td>
</tr>
<tr>
<td>• Works well even when number of input features $n$ is large</td>
<td>• Slow if $n$ is too large (inverting $n \times n$ matrix)</td>
</tr>
</tbody>
</table>
NON-LINEAR FEATURES
(NON-LINEAR BASIS FUNCTIONS)
Non-linear trends in data

- How can we learn non-linear hypotheses?

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>-0.27</td>
</tr>
<tr>
<td>-1.22</td>
<td>2.63</td>
</tr>
<tr>
<td>0.17</td>
<td>-0.13</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

$h_\theta(x) = \theta_0 + \theta_1 \cdot x + \theta_2 \cdot x^2$
Linear fit to this „non-linear“ data

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>-0.27</td>
</tr>
<tr>
<td>-1.22</td>
<td>2.63</td>
</tr>
<tr>
<td>0.17</td>
<td>-0.13</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\[
X = \begin{pmatrix}
1 & 0.01 \\
1 & -1.22 \\
1 & 0.17 \\
\vdots
\end{pmatrix}
\quad y = \begin{pmatrix}
-0.27 \\
2.63 \\
-0.13 \\
\vdots
\end{pmatrix}
\]

*standard design matrix*

Hypothesis: \( h_\theta(x) = \theta_0 + \theta_1 \cdot x \)

Optimal parameters: \( \theta^* = \left( X^T X \right)^{-1} X^T y \)
Linear fit to this „non-linear“ data

\[ h_\theta(x) = 1.85 - 0.76 \cdot x \]
Non-linear (quadratic) fit

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>-0.27</td>
</tr>
<tr>
<td>-1.22</td>
<td>2.63</td>
</tr>
<tr>
<td>0.17</td>
<td>-0.13</td>
</tr>
</tbody>
</table>

$$\phi_0 = 1 \quad \phi_1 = x \quad \phi_2 = x^2$$

$$\Phi = \begin{pmatrix}
1 & 0.01 & 0.01^2 \\
1 & -1.22 & (-1.22)^2 \\
1 & 0.17 & (0.17)^2 \\
\vdots & \vdots & \vdots \\
\end{pmatrix} \quad y = \begin{pmatrix}
-0.27 \\
2.63 \\
-0.13 \\
\vdots \\
\end{pmatrix}$$

design matrix with non-linear features

Hypothesis:
$$h_\theta(\phi) = \theta_0 + \theta_1 \cdot \phi_1 + \theta_2 \cdot \phi_2$$

Optimal parameters:
$$\theta^* = \left(\Phi^T \Phi\right)^{-1} \Phi^T y$$
Non-linear (quadratic) fit

\[
\phi_0 = 1 \\
\phi_1 = x \\
\phi_2 = x^2
\]

\[ h_\theta(x) = 0.02 \cdot 1 - 0.95 \cdot x + 0.99 \cdot x^2 \]
Non-linear (sinusoid) fit

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>-0.27</td>
</tr>
<tr>
<td>-1.22</td>
<td>2.63</td>
</tr>
<tr>
<td>0.17</td>
<td>-0.13</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\[ \Phi = \begin{pmatrix} 1 & 0.01 & \cos(0.01) \\ 1 & -1.22 & \cos(-1.22) \\ 1 & 0.17 & \cos(0.17) \\ \vdots & \vdots & \vdots \end{pmatrix}, \quad y = \begin{pmatrix} -0.27 \\ 2.63 \\ -0.13 \\ \vdots \end{pmatrix} \]

"design matrix with non-linear features"

Hypothesis: \[ h_\theta(\phi) = \theta_0 + \theta_1 \cdot \phi_1 + \theta_2 \cdot \phi_2 \]

Optimal parameters: \[ \theta^* = \left( \Phi^T \Phi \right)^{-1} \Phi^T y \]
Non-linear (sinusoid) fit

\[ h_\theta(x) = 3.12 \cdot 1 - 1.07 \cdot x - 3.5 \cdot \cos(x) \]
Image: JPEG = cosin-basis

Each block of 8x8 pixels is represented in a Fourier basis of cosin filters.

Better representation of edges and corners.

Allows for compression.
Audio: cosin or wavelet basis

Good signal representation make a compromise between time and frequency

\[ f(t) \]

(Mother Wavelet)

\[ \frac{1}{\sqrt{2}} f \left( \frac{t}{2} \right) \]

\[ \frac{1}{2} f \left( \frac{t}{4} \right) \]
Non-linear input features (in general)

Feature 2 for each training example $i$ is computed by applying a non-linear basis function:

$$\phi_2^{(i)} = \phi_2(x^{(i)})$$

Allows to learn a variety of non-linear functions with the same technique(s):

- Gradient descent or

$$\theta^* = \left(\Phi^T \Phi\right)^{-1} \Phi^T y$$
Polynomial regression

- Features are powers of $x$

\[ \phi_0 = x^0, \quad \phi_1 = x^1, \quad \phi_2 = x^2, \ldots, \quad \phi_n = x^n \]

$n = \text{degree of polynome to be learned}$

What happened here? Next lecture…
Radial basis functions

• „Gaussian“-shaped RBFs:
  • Each basis function $j$ has a center $c_j$ in the input space
  • The width of the basis functions is determined by $\sigma$.

$$\phi_j(x) = \exp \left( -\frac{1}{2\sigma^2} \cdot \|x - c_j\|^2 \right)$$

![Graph of Radial Basis Functions](image)
Radial basis functions

- "Gaussian"-shaped RBFs:
  - Each basis function $j$ has a center $c_j$ in the input space
  - The width of the basis functions is determined by $\sigma$.

$$
\phi_j(x) = \exp \left( -\frac{1}{2\sigma^2} \cdot \left\| x - c_j \right\|^2 \right)
$$

![Graph showing Gaussian-shaped RBFs with different centers and widths.](image)

$\sigma = 0.5$

$c_1 = -1 \quad c_2 = 1 \quad c_3 = 3$
Radial basis functions

- "Gaussian"-shaped RBFs:
  - Each basis function $j$ has a center $c_j$ in the input space
  - The width of the basis functions is determined by $\sigma$.

$$
\phi_j(x) = \exp \left( -\frac{1}{2\sigma^2} \cdot \|x - c_j\|^2 \right)
$$

![Graph showing three Gaussian RBFs with different centers and width $\sigma = 1.5$. The centers are $c_1 = -1$, $c_2 = 1$, and $c_3 = 3$.](image)
Fitting a single RBF to data

\[ h_\theta(x) = \theta_0 + \theta_1 \cdot \phi_1(x) \]

\[ h_\theta(x) = 6.9 - 7.3 \cdot \phi_1(x) \]
Fitting RBFs to data

RBFs with $\sigma = 1.5$

$$h_\theta(x) = \theta_0 + \theta_1 \cdot \phi_1(x) + \theta_2 \cdot \phi_2(x) + \theta_3 \cdot \phi_3(x)$$

$$h_\theta(x) = 21.7 - 11.4 \cdot \phi_1(x) - 10.6 \cdot \phi_2(x) - 14.9 \cdot \phi_3(x)$$
SUMMARY (QUESTIONS)
Some questions…

• Hypothesis for linear regression = ?
• Cost function for linear regression = ?
• How many local minima may the cost function for lin. reg. have (under regular conditions)?
• Name two ways to minimize the cost function?
• General gradient descent formula?
• Linear regression with gradient descent formula?
• What issues can arise during gradient descent?
• What is the design matrix? What are its dimensions?
• Analytical solution for linear regression = ?
  • What are the components of the solution?
• Pros and Cons of gradient descent vs. analytical solution?
• How can one learn non-linear hypotheses with linear regression?
• What is polynomial regression?
• What are radial basis functions?
What is next?

• Classification with Logistic Regression

• Gradient descent tricks & more advanced optimization techniques

• Underfitting & Overfitting

• Model selection (Training- & Validation- & Testset)